A Cross-Platform Data Acquisition and Transformation Approach for Whole-Systems Experimentation – Status and Challenges

Jan Moringen, Arne Nordmann, and Sebastian Wrede*

I. Introduction

The emerging availability of data acquisition, processing and analysis tools as well as multi-modal data sets originating from long-term robotics experiments facilitate collaborative and comparative research on Human-Robot Interaction (HRI) at the systems level. A key requirement for sharing and using experimental data within the HRI domain (but also beyond) is openness. Infrastructure for recording and processing experimental data requires standardized protocols and formalization of the syntax and semantics of the underlying data formats.

While these aspects can be specified explicitly for a single system, experiment or associated framework, the development of a more general approach for this problem would be beneficial. Such an approach would not only allow to record data from different sources, e.g., the various robot hard- and software platforms but also external sensors for recording ground truth such as eye or motion tracking devices typically used in HRI research. Furthermore, the integration and data processing with various analysis tools (such as Matlab, R, or annotation tools such as ELAN [6]) would become much easier in this case. In the following, we will briefly describe the current state of a data recording, transformation and processing approach that explicitly targets openness and highlight particular challenges related to this goal.

II. A Toolchain for Cross-Platform Data Acquisition, Transformation and Processing

To facilitate robotics research with repeatable trials and interchangeable data sets, we have implemented generic recording and data processing tools, called RSBag[1]. These tools allow to record system data in the form of event notifications on parts or all communication endpoints of a system, inspect, transform and replay these with different strategies. This includes replay with original and modified speed, interactive stepping and replay as fast as possible. Replaying this data enables the *application of the data set in the integrated system* while ensuring the *availability of annotations* without depending on a concrete annotation tool. Processing data in their original format has several benefits; for example, it allows to evaluate unchanged system components based on the recorded data with their usual inputs like audio and video streams while also having the annotations available in the respective systems at runtime.

*J. Moringen, A. Nordmann, and Sebastian Wrede are with the Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Bielefeld, Germany. jmoringe at cor-lab.uni-bielefeld.de

[1]The name was intentionally chosen with regard to the ROS bag tool.

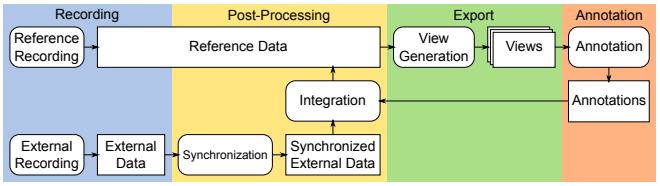

Fig. 1. Schematic overview of whole-system approach for data recording. Rounded boxes indicate performed activities, squares represent generated data. Different workflow phases while creating the data set are indicated through the background colors.

The time handling of all recorded event notifications in the RSBag toolchain is possible, because timing aspects are encoded in a generic structure which is stored alongside with the raw data of all events, independent of the concrete payload. This structure is implemented according to the Time-Indexed Data Entries (TIDE) log file format specification under discussion as RETF RD1. It is itself inspired by the original Bag format, which was developed for logging ROS messages. The TIDE format has been designed for fast streaming of data to disc, fast playback and random access in time of the logged data.

To acquire data from different frameworks, the cross-platform RSB middleware [5] library is used. RSB allows to interface natively with other frameworks such as YARP [2] or ROS [3] to access communicated system data. Its architecture supports this by providing dedicated extension points for interfacing with foreign frameworks and handle different binary encoding rules. In order to configure and select the endpoints, RSB employs URL schemata as exemplified in Table I. We consider encoded payloads as *reference data* that is natively stored by RSBag in the TIDE log files. Moreover, we propose that additional recording devices are either captured directly with this middleware-based recording system or that their data is later integrated into the TIDE log files as shown in Figure 1 facilitating a whole-systems approach [1] for analysis and evaluation. This integration is also the proposed method for *secondary data* like annotations, which are derived from the reference data.

In recent work, we proposed an approach using model-based techniques for improving software component reusability [4]. We specifically addressed data type compatibility through the development of a generic meta-model capable of representing data types from different frameworks and their relations. Based on this model a code generator emits serialization code, which makes it possible to seamlessly reuse the existing data types of different frameworks.

TABLE I

EXEMPLARY RSB URLS USED TO CONFIGURE THE RSBAG TOOLCHAIN FOR RECORDING SYSTEM DATA IN TYPICAL ROBOTICS FRAMEWORKS.

`spread://foo:9999/bar`	Spread daemon listening on port 9999 on host "foo", record all data under scope /bar.
`yarp://icub_Sim/cam/left`	Resolve port "/icub_Sim/cam/left" via YARP nameserver and connect to it using the YARP TCP protocol. Use defaults for nameserver hostname and port.
`yarp://foo:3333/icub_Sim/cam/left`	Like previous example, but connect to YARP nameserver at port 3333 on host "foo".
`yarp+tcp://bar:1000`	Bypass YARP nameserver and directly connect to the YARP port at port number 1000 on host "bar".
`ros+tcp://foo/bar`	Use ROS master to find publishers of topic "/foo/bar" and connect to all using ROS TCP protocol. Use default hostname and port for ROS master.
`<empty string>`	Use all available RSB transports with respective defaults, scope /. For YARP (if enabled), connects to all ports. For ROS (if enabled) connects to all nodes.

This feature is an essential part of the cross-platform data recording and processing toolchain as it allows to handle the specific payloads contained in the logged event notifications using a generic approach. For instance, the generated converters allow to transform payloads recorded in a specific data type A of framework A' to be mapped and replayed as a semantically equivalent data type B utilized in a framework or analysis tool B'. For example, it is easily possible to transform different joint angle representations or image formats to facilitate further analysis.

III. CHALLENGES

The cross-platform data logging approach introduced above poses a number of requirements at the level of the middlewares or device drivers to be integrated. Current robotics frameworks usually lack dedicated mechanisms for type reflection at runtime, which we consider a major problem for providing a cross-platform toolchain for data recording and processing. In particular, features that allow

- unambiguous identification of the used data types,
- lookup of type definitions and further information,
- specification of type definition language and encoding

are usually missing in current robotics frameworks or are implemented in a framework-dependent way.

For instance, probably due to YARP's dynamically typed data structures, this framework does not feature a data definition language which allows type specification and referencing. The RSB middleware communicates a type name but no type definition while ROS transmits type definitions with dependency closure, hash values, and a type name but also lacks explicit version information. For a sustainable handling of experimental data and to allow efficient implementation of communication protocols, a uniform way of referencing robot data type definitions would nevertheless be useful.

Furthermore, logging tools which need to process the data should be able to retrieve type names, information about the used definition language and encoding mechanisms, type definitions including dependency closure, versions and further aspects such as documentation at recording time to store this information as metadata alongside the recorded raw data. While examples like URN or URI schemata exist to unambiguously reference type information for the sake of cross-platform use of recorded data, we are not aware of any toolchain or specification that implements this functionality. A framework independent way of referencing and looking up type definitions could also be beneficial to gather information about the use of data types in a dedicated registry and improve interoperability of robotics software components; for example to provide mappings between different data types or by adding information about these types such as documentation, unit information, etc.

Within the presented approach, a uniform way of handling type definitions would allow us to make full use of the code generation approach by being able to process data from various sources with data type definitions which where not available at release time of the cross-platform data recording and processing framework.

IV. CONCLUSIONS

Given the potential of a generic type referencing and handling scheme and its importance for long-term applicability of recorded experimental data, we are strongly interested in a discussion on how to setup the development of a framework-independent mechanism and registry for robotics data type definitions. We believe that this would be beneficial for the community and could be realized with only little administrative effort, e.g., through a service provided by the Robot Engineering Task Force.

REFERENCES

[1] Manja Lohse, Marc Hanheide, Katharina Rohlfing, and Gerhard Sagerer. Systemic Interaction Analysis (SInA) in HRI. In *Proc. Int. Conf. Human-Robot Interaction*, 2009.

[2] Giorgio Metta and Paul Fitzpatrick. YARP: yet another robot platform. *Journal on Advanced Robotics*, 3(1):43–48, 2006.

[3] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *International Conference on Robotics and Automation*, number Figure 1, 2009.

[4] J. Wienke, A. Nordmann, and S. Wrede. A Meta-Model and Toolchain for Improved Interoperability of Robotic Frameworks. In *Simulation, Modeling, and Programming for Autonomous Robots*, 2012.

[5] Johannes Wienke and Sebastian Wrede. A Middleware for Collaborative Research in Experimental Robotics. In *IEEE/SICE International Symposium on System Integration (SII2011)*, Kyoto, Japan, 2011. IEEE.

[6] P. Wittenburg et al. Elan: a professional framework for multimodality research. In *Proc. of LREC 2006, Fifth Int. Conf. on Language Resources and Evaluation*, 2006.